AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions of claims in the application:

Listing of Claims:

1. (Previously presented) A language-neutral representation of a compile unit transformable to at least one of a plurality of different types of code representations, the language-neutral representation of the compile unit resident on one or more computers and comprising:

a hierarchal arrangement of program elements that neutrally characterize the compile unit;

at least one of the program elements representing a type declaration that characterizes at least one class of programmatic constructs of the compile unit; and

a code generator that converts the language neutral representation of the compile unit to a corresponding representation of the compile unit in at least one high-level language code.

2. (Previously presented) The language-neutral representation of claim 1, further comprising a collection of at least one member that characterizes programmatic attributes associated with and able to be implemented within the at least one class.

3. (Previously presented) The language-neutral representation of claim 2, wherein the collection further comprises an expression class within the at least one class.

4. (Previously presented) The language-neutral representation of claim 2, wherein the collection further comprises a statement class within the at least one class.

5. (Previously presented) The language-neutral representation of claim 2, wherein the hierarchal arrangement further comprises a namespace that contains the at least one class.

6. (Previously presented) The language-neutral representation of claim 1, wherein at least one of the program elements of the hierarchal arrangement encapsulates another of the program elements.

7. (Previously Presented) The language-neutral representation of claim 1 in combination with an interface associated with the language-neutral representation, the interface transforms the language-neutral representation to a corresponding desired code representation.

8. (Previously Presented) The language-neutral representation of claim 7, wherein the program elements comprise objects, each object exposing at least one of a method, attribute, and property of each respective object, the interface employs the at least one of method, attribute and property to facilitate the transformation into the desired code representation.

9. (Previously presented) The language-neutral representation of claim 7, wherein the interface further comprises a compiler interface programmed to transform the language-neutral representation to a corresponding representation in a low-level language code.

10. (Previously presented) The language-neutral representation of claim 9, wherein the representation in a low-level language code further comprises an assembly of computer-executable instructions.

11. (Previously presented) The language-neutral representation of claim 7, wherein the interface further comprises a code generator interface programmed to convert the language-neutral representation to a plurality of corresponding representations, wherein each representation is in a different high-level language code.

12. (Previously presented) The language-neutral representation of claim 1, wherein the program elements comprise instances of a plurality of language-neutral classes, each instance defining an associated object.

13. (Previously presented) The language-neutral representation of claim 12, wherein at least one associated object represents the type declaration, at least another object being encapsulated within the at least one object representing the at least one type declaration, the at least another object representing program code of the compile unit that derives from a class associated with the at least type declaration.

14. (Previously presented) A language-neutral representation of programmatic elements resident upon a computer-readable medium, comprising:

an instance of at least one of a plurality of language-neutral classes, the plurality of classes representing different programmatic constructs of a compile unit and having a hierarchal relationship relative to each other, whereby transformation of the instance into a different representation of the respective programmatic construct is facilitated; and

a code generator that converts the instance of at least one of a plurality of language-neutral classes into a corresponding different representation of the instance in at least one high-level language code.

15. (Previously presented) The language-neutral representation of claim 14, further comprising a plurality of instances of the plurality of the classes, wherein each instance of a corresponding class of the plurality of classes represents a respective programmatic construct of the compile unit, the plurality of instances being organized in a hierarchal relationship based on the classes associated with the plurality of instances and relationships among the programmatic constructs represented thereby.

16. (Previously presented) The language-neutral representation of claim 15, wherein each of the plurality of instances exposes at least one item associated with the programmatic construct represented thereby.

17. (Previously presented) The language-neutral representation of claim 16, wherein at least one of the plurality of instances represents a type declaration, at least another instance being encapsulated within the instance representing the type declaration, the at least another instance representing a programmatic construct that derives from the at least type declaration.

18. (Previously presented) The language-neutral representation of claim 17 wherein the at least another object further comprises at least one of a statement and an expression.

19. (Previously presented) The language-neutral representation of claim 16 in combination with an interface that transforms the language neutral representation to the different representation, the interface employs the at least one item to facilitate the transformation of the language-neutral representation into the different representation.

20. (Previously presented) The language-neutral representation of claim 19, wherein the interface further comprises a compiler interface programmed to transform the language-neutral representation to the corresponding different representation in a low-level language code.

21. (Previously presented) The language-neutral representation of claim 20, wherein the different representation in a low-level language code further comprises an assembly of computer-executable instructions.

22. (Previously presented) The language-neutral representation of claim 19, wherein the interface further comprises a code generator interface programmed to generate a plurality of corresponding representations from the language-neutral representation, wherein each representation is in a different high-level language code from the language-neutral representation.

23. (Previously presented) A language-neutral representation of computer executable instructions that are transformable to at least one other type of software code representation, the language-neutral representation existent within a computer-readable medium, comprising:

a hierarchal arrangement of objects, each object representing a different program element of the compile unit;

at least one class object that represents at least one defined class of program elements of the compile unit;

at least one member object associated with the at least one class object that represents computer-executable instructions operable on at least some program elements in the at least one defined class; and

a code generator that converts the language-neutral representation of computer executable instructions into a corresponding representation in at least one high-level language code.

24. (Previously presented) The language-neutral representation of claim 23, further comprising a namespace object that represents a namespace of the compile unit, the namespace object comprising a collection of class objects including the at least one class object.

25. (Previously presented) The language-neutral representation of claim 24, further comprising a plurality of member objects associated with the at least one class object, wherein the at least one class object represents a common base class that is shared by the plurality of member objects.

26. (Previously presented) The language-neutral representation of claim 24, wherein the plurality of member objects further comprise a collection of objects representing at least one of a statement and an expression of the compile unit.

27. (Previously Presented) The language-neutral representation of claim 23 in combination with an associated interface, the interface transforms the language-neutral representation to a corresponding desired code representation.

28. (Previously Presented) The language-neutral representation of claim 27, wherein each object exposes at least one item indicative of the program element represented thereby, the interface employs each exposed item to facilitate transformation of the language-neutral representation into the desired code representation.

29. (Previously presented) The language-neutral representation of claim 28, wherein the interface further comprises a compiler interface that exposes computer-executable instructions to transform the language-neutral representation to a corresponding representation in a low-level language code.

30. (Previously presented) The language-neutral representation of claim 29, wherein the representation in a low-level language code further comprises an assembly of computer-executable instructions.

31. (Previously presented) The language-neutral representation of claim 28, wherein the interface further comprises a code generator interface that exposes computer-executable instructions to convert the language-neutral representation to a plurality of corresponding representations, wherein each representation is in a different high-level language code.

32. (Previously presented) A language-neutral representation of programmatic elements within a computer-readable medium, comprising:

means for representing different code portions of a compile unit in a language-neutral manner, the means for representing being arranged according to a hierarchy;

means for converting the language neutral representation of the compile unit into a representation of the compile unit in a low level language code; and

means for converting the language neutral representation of the compile unit into a representation of the compile unit in a high level language code.

33. (Previously presented) The language-neutral representation of claim 32, further comprising means for transforming the language-neutral representation to a plurality of corresponding desired representations of code.

34. (Previously presented) The combination of claim 33, wherein the means for transforming further comprises means for compiling the language-neutral representation to a plurality of corresponding representations, wherein each representation is in a different low-level language code representation.

35. (Previously presented) The language-neutral representation of claim 33, wherein the means for transforming further comprises means for generating a plurality of corresponding representations from the language-neutral representation, wherein each representation is in a different high-level language code.

36. (Previously Presented) A computer implemented system that transforms a language-specific representation of a compile unit to a language-neutral representation thereof, the system comprising the following computer executable components:

a plurality of language-neutral classes that represent different programmatic constructs and have a hierarchal relationship relative to each other;

an interface that exposes the plurality of classes;

a design system that employs the interface to instantiate selected classes of the plurality of classes to create corresponding objects that represent respective code elements of the compile unit arranged according to the hierarchal relationship to define the language-neutral representation; and

a code generator that converts the language-neutral representation of the compile unit into a corresponding representation in at least one high-level language code.

37. (Previously presented) system of claim 36, wherein at least one of the objects is a namespace object that comprises a collection of at least one base class member object that provides a common base class for at least some of the objects.

38. (Previously presented) The system of claim 37, wherein the at least one base class further comprises at least one member comprising a collection of objects representing at least one of a statement and an expression of the compile unit.

39. (Previously presented) The system of claim 36, wherein the code generator interface is programmed to transform the language-neutral representation to a high-level language code that is different from the language-specific representation.

40. (Previously presented) A computer-readable medium having stored thereon computer-executable instructions for creating a language-neutral representation of a compile unit, comprising:

a language-neutral representation comprising associated objects that represent different parts of the compile unit arranged according to an established hierarchy, each of the objects being an instance of a respective class of a plurality of language-neutral classes, each respective class defining a different part of code according to the established hierarchy; and

a code generator that converts the language-neutral representation of the compile unit into a corresponding representation in at least one high-level language code.

41. (Previously Presented) A method to transform a representation of a compile unit in a first high level language code to a language-neutral representation thereof, the method comprising:

mapping each of a plurality of programmatic constructs of the first high level language code to a corresponding class of a plurality of language-neutral classes, the classes having a hierarchal relationship relative to each other;

instantiating each corresponding class based on the mapping to create corresponding objects that represent respective programmatic constructs of the compile unit;

arranging the objects according to the hierarchal relationship to define the language neutral representation; and

converting the language-neutral representation of the compile unit into a corresponding representation in at least one second high-level language code, the second high level language code is in a different high level language than the first high level language code.

42. (Previously Presented) The method of claim 41, further comprising transforming the language-neutral representation into a corresponding representation of the compile unit in a high-level language code that is the same as the representation of the compile unit in the first high level language code.

43. (Previously Presented) A computer implemented system to transform a language-neutral representation of a compile unit to a high-level language, comprising the following computer executable components:

an interface that controls manipulation of a plurality of language-neutral program elements of the compile unit, each of the plurality of program elements being an instance of a corresponding class of a plurality of classes arranged according to a hierarchy; and

a code generator that implements the interface to transform each of the program elements to a representation in a high-level language code thereof corresponding to program information associated with each respective program element.

44. (Previously presented) The system of claim 43, wherein the interface further comprises a class interface to expose program information relating to a base class of the compile unit representing at least one type declaration.

45. (Previously Presented) The system of claim 44, wherein the interface further comprises a member interface that exposes program information relating to a member of the base class of the compile unit.

46. (Previously presented) The system of claim 43, wherein the member of the base class represents a base class for at least one of a statement and an expression.

47. (Previously presented) A method to transform a language-neutral representation of a compile unit to a corresponding target language-based representation, the language-neutral representation including at least one language-neutral program element, the at least one program element being an instance of a corresponding class of a plurality of classes arranged according to a hierarchy, the method comprising:

providing an interface to expose at least one method operative to control at least one of manipulation of program elements and compilation of the language neutral representation; and

depending on the target language-based representation, converting the at least one program element to a representation in a high-level language code thereof according to the at least one exposed method or implementing the interface relative to the language neutral representation to compile each instance of the language-neutral representation into a representation in a low-level language code.

48. (Previously Presented) A computer implemented system to translate a language-neutral representation of a compile unit to a low-level language, comprising the following computer executable components:

an interface that exposes methods to control compilation of the language-neutral representation;

a compiler that implements the interface to translate a plurality of program elements that define the language-neutral representation into the low-level language code, each of the plurality of program elements being an instance of a corresponding class of a plurality of classes that represent program constructs, the plurality of elements being arranged according to a hierarchy; and a code generator that converts the plurality of program elements that define the language-neutral representation of the compile unit into a corresponding representation in at least one high-level language code.

49. (Previously presented) The system of claim 48, wherein the low-level language code comprises at least one of an assembly, byte code and intermediate language.